# Glowworm swarm optimisation for training multi-layer perceptrons

Alboaneen, Dabiah Ahmed; Tianfield, Huaglory; Zhang, Yan

*Published in:*
Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT '17)

*DOI:*
10.1145/3148055.3148075

*Publication date:*
2017

*Document Version*
Author accepted manuscript

# Glowworm Swarm Optimisation for Training Multi-Layer Perceptrons

Dabiah Ahmed Alboaneen*
Department of Computer, Communications and Interactive Systems
Glasgow Caledonian University
Glasgow, UK
dabiah.alboaneen@gcu.ac.uk

Huaglory Tianfield
Department of Computer, Communications and Interactive Systems
Glasgow Caledonian University
Glasgow, UK
h.tianfield@gcu.ac.uk

Yan Zhang
Department of Computer, Communications and Interactive Systems
Glasgow Caledonian University
Glasgow, UK
yan.zhang@gcu.ac.uk

## ABSTRACT

Training multi-layer perceptron (MLP) is non-trivial due to its non-linear nature and the presence of large number of local optima. Meta-heuristic algorithms may solve this problem efficiently. In this paper, we investigate the use of glowworm swarm optimisation (GSO) algorithm in training the MLP neural network. The GSO based trainer is evaluated on five classification datasets, namely Wisconsin breast cancer, BUPA liver disorders, vertebral column, exclusive OR (XOR) and balloons. The evaluations are conducted by comparing the proposed trainer with four other meta-heuristics, namely biogeography-based optimisation (BBO), genetic algorithm (GA), bat (BAT) and multi-verse optimiser (MVO) algorithms. The results show that our proposed trainer achieves better classification accuracy rate in most datasets compared to the other algorithms.

## CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Theory of computation** → **Bio-inspired optimization**;

## KEYWORDS

Multi-layer perceptrons; neural network; glowworm swarm optimisation (GSO)

## 1 INTRODUCTION

Multi-layer perceptron (MLP) is a type of feed-forward artificial neural networks (ANNs) [32]. In machine learning algorithms, the training process is an important step and it can affect the performance of the neural networks. The purpose of training MLP networks is to find the best set of connection weights to minimise the prediction (classification or approximation) error. Gradient-based algorithms such as back-propagation (BP) are considered to be a conventional choice for MLP training process [17]. However, for complex problems, gradient-based algorithms suffer from high dependency on the initial solution, high probability of local optima stagnation [13] [21], and slow convergence [9].

The drawbacks of gradient-based algorithms in training MLP networks motivate researchers to turn to different meta-heuristic algorithms as an alternative to gradient-based algorithms for training MLP networks. In addition, it has been validated by the no free lunch (NFL) theorem that there is no heuristic algorithm that is able to solve all optimisation problems [33] [15].

Glowworm swarm optimisation (GSO) was introduced by Krishnan and Ghose in 2006 [20] and is inspired by the social behaviour of glowworm. It is based on a swarm of glowworms moving through search space and communicating with each other in order to determine a search direction. GSO has fewer parameters to tune, which makes it easier to implement. This paper will apply GSO to train an MLP network, i.e., to optimise connection weights and biases.

The remainder of this paper is arranged as follows. Section 2 presents literature review on training MLPs using meta-heuristic algorithms. A brief preliminary on MLP and GSO is given in Section 3. Section 4 puts forward GSO-based MLP training. Experimental evaluations are discussed in section 5. Finally, Section 6 draws conclusion and sets future work.

## 2 LITERATURE REVIEW

Existing studies on meta-heuristic algorithms for training MLPs can be divided into two categories. (i) To find an appropriate structure or adjust the parameters for an MLP in a specific problem and (ii) to optimise weights and biases that provide the minimum classification error for an MLP.

### 2.1 Optimising Structure and Parameters

GA was employed to define the structure of ANN and to tune the parameters of an ANN [29] [22]. Particle swarm optimisation (PSO) was employed to define the structure of MLP [37]. In [6], simulated annealing (SA) was combined with BP for optimising MLP structure and adjusting its connection weights. It's showed that the SA based trainer can build an MLP with sufficient number of hidden neurons that satisfy performance. In addition, its performance was better than the GA for the same purpose.

In [39], a hybrid approach combining SA, tabu search (TS), GA and BP, called GaTSa, was proposed to optimise the structure and weights of the MLPs. GaTSa trainer can add new neurons in the structure based on GA, escape from local minima (SA feature) and

*The author is academic staff with the department of computer, University of Imam Abdulrahman Bin Faisal, Jubail, Saudi Arabia.

achieve fast convergence by the evaluation of a set of solutions (TS feature). It's showed that GaTSa outperforms the other methods for most problems.

In [38], cat swarm optimisation (CSO) algorithm and optimal brain damage (OBD) pruning technique were used to simultaneously optimise the connection weights and MLP structure. CSO optimiser trains MLP to learn the input-output relationships of a given problem and then uses the OBD pruning method to generate an optimal network structure. It's showed that a CSO optimiser with OBD pruning algorithm was able to generate an optimal set of connection weights and MLP structure for all datasets with low training error and high classification accuracy.

## 2.2 Optimising Weights and Biases

Meta-heuristic algorithms have been widely applied for optimising weights and biases of MLPs. GA is one of the first meta-heuristics used to train MLPs network [30]. Optimising weights and biases of MLPs for classification problems has been done by using artificial bee colony (ABC) [31], artificial fish swarm (AFS) [14], magnetic optimisation algorithm (MOA) [27], cuckoo search (CS) [18], firefly algorithm (FA) [8] [5], BBO [26], grey wolf optimiser (GWO) [23], chaotic shark smell optimisation (CSSO) [1], moth-flame optimiser (MFO) [35], social spider optimisation (SSO) [28], MVO [11], whale optimisation algorithm (WOA) [4] and symbiotic organisms search (SOS) [34].

Hybrid algorithms to enhance the weights and biases of MLP have also been proposed and evaluated. In [24], a hybrid of PSO and gravitational search algorithm (GSA) was employed to train MLP network in order to address the problems of trapping in local minima and slow convergence rate. It's showed that PSOGSA outperforms both PSO and GSA for training MLPs in terms of converging speed and local minima avoidance and it has better accuracy than GSA. In [10], improved monarch butterfly optimisation (IMBO) algorithm was employed to search for the connection weights and biases that minimise the prediction error of the network. It's showed that IMBO improves the training of MLPs. IMBO provided fast convergence compared to other meta-heuristic algorithms in most of the datasets. In addition, it achieved better accuracy rates than most of other algorithms.

Using meta-heuristic algorithms for training different types of neural networks was investigated in [19]. Three types of neural networks were trained by using the modified PSO (MPSO) to improve the classification performance. It's showed that the MPSO algorithm can improve the classification performance for the three neural network types.

# 3 PRELIMINARIES

## 3.1 Multi-Layer Perceptron

A MLP network starts with an input layer followed by hidden layers and ends with an output layer. Hidden layers provide the computational processing in the network to produce the network outputs. Figure 1 illustrates an example of MLP with one hidden layer. The connections between the layers are called weights $W$, which are normally defined between 0 and 1. The output value of each neuron in each layer is calculated in two stages as below.

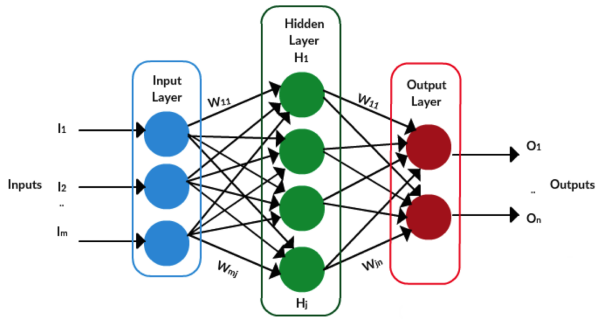



**Figure 1: MLP structure with $m$ inputs, one hidden layer and $n$ outputs.**

In the first stage, the weighted summation of the input values is calculated as below:

$$\forall l \in \{1, 2, ..., j\}, h_l = \sum_{i=1}^{m} W_{il} I_i + \beta_l \quad (1)$$

where $I_i$ is the input variable $i$, $W_{il}$ is the connection weight between $I_i$ and the hidden neuron $l$, $m$ is the total number of inputs and $\beta_l$ is the bias of the $l^{th}$ hidden neuron.

In the second stage, the output value of each neuron in the hidden layer is calculated based on a weighted summation using an activation function, e.g. the sigmoid activation function, to map the hidden layer to output values. That is,

$$\forall l \in \{1, 2, ..., j\}, H_l = sigmoid(h_l) = \frac{1}{1 + e^{-h_l}} \quad (2)$$

The final output of the network is calculated as below:

$$\forall k \in \{1, 2, ..., n\}, o_k = \sum_{l=1}^{j} W_{lk} H_l + \beta_k \quad (3)$$

$$\forall k \in \{1, 2, ..., n\}, O_k = sigmoid(o_k) = \frac{1}{1 + e^{-o_k}} \quad (4)$$

where $W_{lk}$ is the connection weight between the $l^{th}$ hidden neuron and the $k^{th}$ output neuron. $\beta_k$ is the bias of the $k^{th}$ hidden neuron.

## 3.2 Glowworm Swarm Optimisation

GSO is based on the behaviour of glowworms. A glowworm that produces more light (high luciferin) means that it is closer to an actual position and has a high objective function value. A GSO algorithm comprises four phases, i.e., initialisation, luciferin updating, moving and local radial range updating. The GSO algorithm can be formulated as in Algorithm 1 [2] [3]. GSO algorithm starts by positioning glowworms randomly in the search space and all the glowworms contain an equal quantity of luciferin. Each glowworm $y$ converts the objective function value $f(x_y(t+1))$ at its current location $x_y(t)$ to a luciferin value $\ell_y(t+1)$ by using the formula below.

$$\ell_y(t+1) = (1-p)l_y(t) + \gamma f(x_y(t+1)) \quad (5)$$

where $\ell_y(t)$ is the luciferin value of glowworm $y$ at time $t$, $p$ is the luciferin decay coefficient ($0 < p < 1$) and $\gamma$ is the luciferin enhancement coefficient.

**Algorithm 1:** GSO: Glowworm Swarm Optimisation

Initialise parameters $\beta$ , $p$, $s$, $z_t$
$\forall y$ , set $\ell_y(0) = \ell_0$
$\forall y$ , set $\gamma_d^y(0) = \gamma_0$
**while** *termination condition not met* **do**
  **for** $y \in m$ **do**
    $\ell_y(t+1) = (1-p)\ell_y(t) + \gamma f(x_y(t+1))$
    $Z_y(t) = \{z : ||x_z(t) - x_y(t)|| \leq \gamma_d^y(t); \ell_y(t) < \ell_z(t)\}$
    **for** *each* $z \in Z_y(t)$ **do**
      $p_{yz}(t) = \frac{\ell_z(t) - \ell_y(t)}{\sum_{w \in Z_y(t)} \ell_w(t) - \ell_y(t)}$
    $x_y(t+1) = x_y(t) + s\left(\frac{x_z(t) - x_y(t)}{||x_z(t) - x_y(t)||}\right)$
    $\gamma_d^y(t+1) = \min\{\gamma_s, \max\{0, \gamma_d^y(t) + \beta(z_t - |Z_y(t)|)\}\}$
  $t = t + 1$
**return** Optimal Solution

Then, each glowworm chooses to move toward one of its neighbours $z$ , using probability, that has a higher luciferin value within the local radial range $\gamma_d$.

$$Z_y(t) = \{z : ||x_z(t) - x_y(t)|| \leq \gamma_d^y(t); \ell_y < \ell_z(t)\} \tag{6}$$

where $Z_y(t)$ is the neighbour set, $z$ is the index of glowworm close to $y$, $x_z(t)$ and $x_y(t)$ are locations of glowworm $z$ and glowworm $y$, respectively, $\ell_y(t)$ and $\ell_z(t)$ are luciferin values for glowworm $y$ and glowworm $z$, respectively. $||x||$ is the Euclidean norm of $x$, and $\gamma_d^y(t)$ represents the local radial range.

$$p_{yz}(t) = \frac{\ell_z(t) - \ell_y(t)}{\sum_{s \in Z_y(t)} \ell_w(t) - \ell_y(t)} \tag{7}$$

where $p_{yz}(t)$ is the probability of glowworm $y$ moving to glowworm $z$.

$$x_y(t+1) = x_y(t) + s\left(\frac{x_z(t) - x_y(t)}{||x_z(t) - x_y(t)||}\right) \tag{8}$$

where $x_y(t+1)$ and $x_y(t)$ are the new and current locations of glowworm, respectively and $s$ is the size of moving step.

Finally, the local radial range $\gamma_d^y$ is updated as below in order to formulate the neighbour set.

$$\gamma_d^y(t+1) = \min\{\gamma_s, \max\{0, \gamma_d^y(t) + \beta(z_t - |Z_y(t)|)\}\} \tag{9}$$

where $\beta$ is the change rate of the neighbourhood range.

## 4 GSO-BASED MLPS TRAINING

This section puts forward the process of using GSO algorithm as a trainer for MLP network of one hidden layer. The MLP with initial settings is employed first to obtain the initial solution and then GSO optimises the weights and biases to minimise the classification error rate of the MLP. Two aspects have been taken into account when designing the approach: (i) the encoding scheme of the search agents in the GSO and (ii) the fitness function.

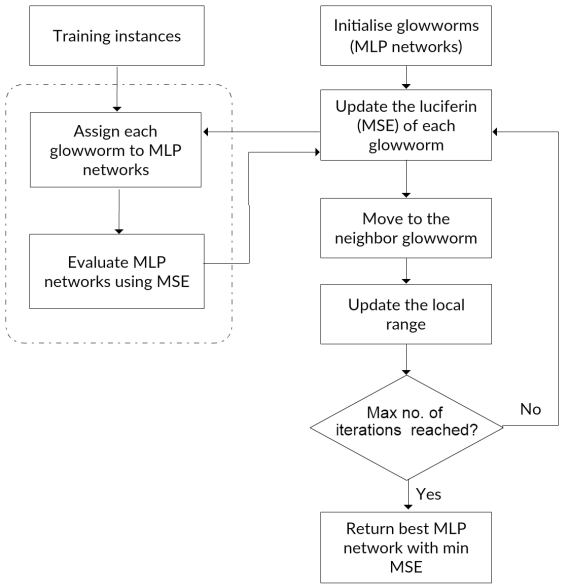



**Figure 2: Flowchart of the GSO-MLP approach.**

(1) **Encoding Scheme:** Each glowworm (individual) in GSO is encoded as a vector of real numbers in the range $[0, 1]$ to represent a candidate MLP network. Vectors include three parts: the connection weights between the input layer and the hidden layer, the connection weights between the hidden layer and the output layer and the biases. The dimension $D$ of the problem (the length of each vector equals the total number of weights and biases in the network) can be calculated as shown below.

$$D = (m * j) + (2 * j) + j \tag{10}$$

where $m$ represents the number of input variables (features) in the dataset, $j$ is the number of neurons in the hidden layer.

(2) **Fitness Function:** Each glowworm is evaluated according to its fitness. This evaluation is done by passing the vector of weights and biases to MLP; then the MSE criterion is calculated based on the difference between the actual and predicted values by the generated agents (MLPs) for all training instances. After the maximum number of iterations is met, the optimal solution is finally achieved, which is regarded as the weights and biases of a MLP network. The aim is to minimise the value of MSE below.

$$MSE = \sum_{t=1}^{T} \frac{\sum_{k=1}^{n} (o_k^t - d_k^t)^2}{T} \tag{11}$$

where $T$ is the total number of instances in the training dataset, $n$ is the total number of outputs, $o_k^t$ is the actual output of the $k^{th}$ input when the $t^{th}$ training instance is used and $d_k^t$ is the desired output of the $k^{th}$ input when the $t^{th}$ training instance is used.

**Table 1: Description of Datasets**

| Dataset | #Instances | #Training instances | #Test instances | #Features | MLP structure |
|---|---|---|---|---|---|
| Wisconsin breast cancer | 699 | 461 | 238 | 8 | 8-17-1 |
| BUPA liver disorders | 345 | 227 | 118 | 6 | 6-13-1 |
| Vertebral column | 310 | 204 | 106 | 6 | 6-13-1 |
| 3-bit XOR | 16 | 16 | 16 | 3 | 3-7-1 |
| Balloons | 20 | 20 | 20 | 4 | 4-9-1 |

Figure 2 shows the flowchart of GSO based trainer. The process of the GSO algorithm for training the MLP network can be summarised in the following steps:

Step 1. A pre-defined number of glowworms are randomly initialised. Each glowworm represents a candidate MLP network.

Step 2. The MLP network is evaluated using a fitness function (MSE).

Step 3. The luciferin of glowworm is updated.

Step 4. Each glowworm moves to the neighbour glowworm.

Step 5. The local range of glowworm is updated.

Step 6. Steps 3 to 5 are repeated until the maximum number of iterations is reached. Finally, the MLP network with the minimum MSE value is validated on test dataset.

## 5 EXPERIMENTAL EVALUATIONS

### 5.1 Datasets

The efficiency of the GSO algorithm for training an MLP network is tested on five datasets. Datasets are obtained from the university of California at Irvine (UCI) machine learning repository [7]. The datasets are listed in Table 1. Each dataset is divided into two parts: 66% of the dataset is used as training data and the remaining 34% is used as testing data [27] [24] [26] [23] [28] [11] [4] [10]. The results are compared to four meta-heuristics algorithms namely, biogeography-based optimisation (BBO), genetic algorithm (GA) [16], bat (BAT) [36] and multi-verse optimiser (MVO) [25] algorithms.

*Wisconsin Breast Cancer.* The dataset has 699 instances, 599 training instances and 100 test instances. Each instance represents a patient that had undergone surgery for breast cancer. It has 8 features and the output is equal to 0 for benign or 1 for malignant cancers.

*BUPA Liver Disorders.* The dataset has 345 instances, 227 training instances and 118 test instances. It includes 6 features and the output is equal to 1 in case of a positive test for the liver disorder or 0 in a negative test.

*Vertebral Column.* The dataset has 310 instances, 204 training instances and 106 test instances. It consists of 6 features used to classify orthopaedic patients into two classes (normal or abnormal).

*Exclusive-OR (XOR).* The N-bit XOR problem is a non-linear classification problem. 3-bits XOR mapping three binary inputs to a single binary output. The problem is to recognise the number of 1 in the input vector. The XOR output should be equal 1 if the input vector contains an odd number of '1's. Otherwise, if the input vector contains an even number of '1's, the output will be 0.

**Table 2: Parameter Settings**

| Algorithm | Parameter | Value |
|---|---|---|
| GSO | Luciferin decay coefficient | 0.4 |
| | Luciferin enhancement coefficient | 0.6 |
| | Rate of the neighbourhood range | 0.08 |
| | No. of neighbours | 5 |
| | Step size of moving | 0.3 |
| | Initial luciferin | 5 |
| BBO | Mutation probability | 0.05 |
| | Number of elites | 2 |
| GA | Crossover probability | 0.8 |
| | Mutation probability | 0.2 |
| BAT | Loudness | 0.5 |
| | Pulse rate | 0.5 |
| | Frequency minimum | 0 |
| | Frequency maximum | 1 |
| MVO | Travelling distance rate | [1, 0.2] |
| | Exploitation accuracy | 6 |

*Balloons.* This dataset is based on different conditions of an experiment in blowing up a balloon. It has 16 instances with 4 features. The dataset output is a binary number indicates whether the balloon is inflated or not.

### 5.2 Algorithm Setup

For all experiments, we used Python to implement the proposed GSO, BBO and GA trainers. For BAT and MVO algorithms, we modified EvoloPy open source [12]. All experiments are executed for 10 different runs to minimise the influence of random effects and to ensure that the results are statistically acceptable. Four meta-heuristic algorithms, including BBO, GA, BAT and MVO are presented for comparison. The parameter settings for all algorithms are shown in Table 2.

The number of inputs is equal to the number of features of each dataset and the number of outputs is equal to the number of classes. One hidden layer is considered. There is no rule to determine the optimal number of neurons. In our experiments we follow the method in the literature [27] [24] [26] [23] [28] [11] [4] [10]. That is, the number of neurons in the hidden layer is $2 * m + 1$, where $m$ is the number of inputs (features) in the datasets. Therefore, the resulted MLP structure for each dataset is illustrated in the last column in Table 1. The population size of all datasets is 50 and the maximum number of iterations is 200.

**Table 3: MSE of Algorithms over Datasets**

| Dataset | | BBO-MLP | GA-MLP | BAT-MLP | MVO-MLP | GSO-MLP |
|---|---|---|---|---|---|---|
| Wisconsin breast cancer | AVG | **0.0386** | 0.0345 | 0.0658 | 0.0430 | 0.0400 |
| | STD | 0.0009 | 0.0016 | 0.0277 | 0.0034 | **0.0007** |
| | BST | 0.0369 | **0.0325** | 0.0346 | 0.0358 | 0.0393 |
| BUPA liver disorders | AVG | 0.4137 | 0.3921 | 0.2126 | 0.2161 | **0.2082** |
| | STD | 0.0664 | 0.0285 | 0.0188 | 0.0060 | **0.0056** |
| | BST | 0.3348 | 0.3392 | 0.1988 | 0.2074 | **0.1966** |
| Vertebral column | AVG | 0.3225 | 0.2186 | 0.1584 | 0.1565 | **0.1457** |
| | STD | 0.0187 | 0.0246 | 0.0275 | 0.0067 | **0.0054** |
| | BST | 0.2941 | 0.1863 | **0.1266** | 0.1457 | 0.1379 |
| 3-bits XOR | AVG | 0.3500 | 0.2500 | 0.1478 | 0.2206 | **0.0381** |
| | STD | 0.1149 | **0** | 0.1111 | 0.0188 | 0.0371 |
| | BST | 0.2500 | 0.2500 | 0.0048 | 0.1828 | **0.0014** |
| Balloons | AVG | 0.1950 | **0** | 0.1870 | 0.0530 | 0.0167 |
| | STD | 0.1343 | **0** | 0.0820 | 0.0099 | 0.0089 |
| | BST | **0** | **0** | 0.0049 | 0.0384 | 0.0040 |

**Table 4: Classification Accuracy of Algorithms over Datasets**

| Dataset | | BBO-MLP | GA-MLP | BAT-MLP | MVO-MLP | GSO-MLP |
|---|---|---|---|---|---|---|
| Wisconsin breast cancer | AVG | **97.5%** | 97.3% | 64.5% | 96.9% | 97.2% |
| | STD | **0.0052** | 0.0093 | 0.3301 | 0.0060 | **0.0052** |
| | BST | **98.3%** | **98.3%** | 97.9% | 97.5% | 97.9% |
| BUPA liver disorders | AVG | 56.6% | 56.7% | 67.8% | 68.2% | **72.8%** |
| | STD | 0.0579 | **0.0129** | 0.1151 | 0.0423 | 0.0162 |
| | BST | 65.3% | 58.5% | **77.1%** | 73.7% | 74.6% |
| Vertebral column | AVG | 68.5% | 71.5% | 59.6% | 79.2% | **83.1%** |
| | STD | 0.0256 | **0.0217** | 0.2690 | 0.0351 | 0.0470 |
| | BST | 70.8% | 75.5% | 87.7% | 84% | **89.6%** |
| 3-bits XOR | AVG | 65% | 75% | 77.5% | 68.8% | **98.8%** |
| | STD | 0.1149 | **0** | 0.2415 | 0.1062 | 0.0395 |
| | BST | 75% | 75% | **100%** | 87.5% | **100%** |
| Balloons | AVG | 80.5% | **100%** | 53.5% | 99% | **100%** |
| | STD | 0.1343 | **0** | 0.2096 | 0.0211 | **0** |
| | BST | **100%** | **100%** | **100%** | **100%** | **100%** |

## 5.3 Results And Discussions

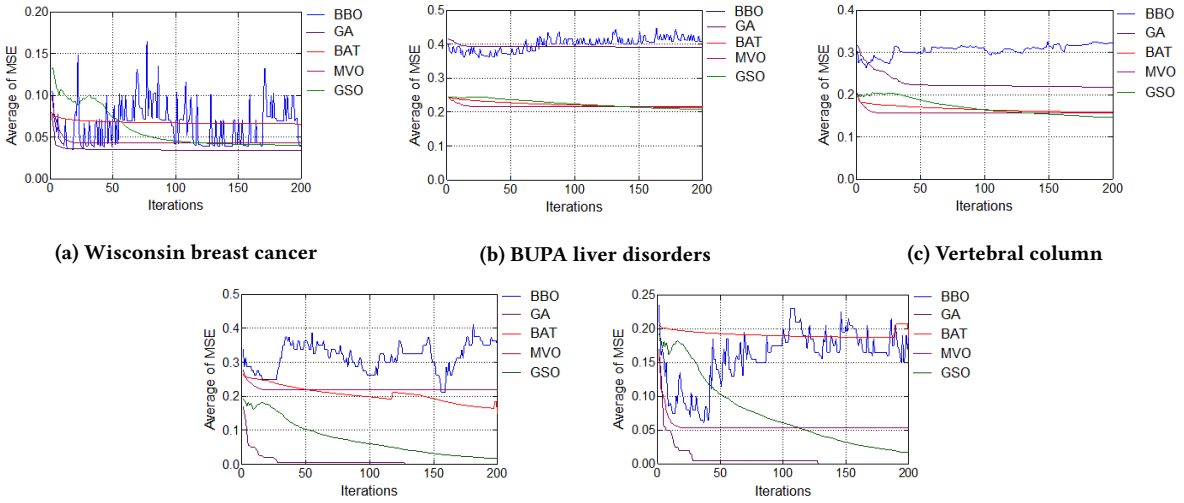
To measure the performance of the GSO based trainer, two parameters have been considered namely; fitness function value (MSE) and classification accuracy rates. We consider the average (AVG), standard deviation (STD) and the best (BST) values of fitness function and classification accuracy rates.

For MSE, AVG indicates the average of MSE over 10 runs, so a lower average value is evidence of an algorithm more successfully avoiding local optima and finding solutions near the global optimum. However, considering the AVG only is not enough because two algorithms can have equal averages, but have different performance in terms of finding the global optimum in each run. Therefore, STD can help to determine how the results are close to each other and reflect the spread. The lower the STD, the lower the dispersion of results. Moreover, BST value is the lowest MSE value that each trainer has achieved over 10 runs.

In addition, the average classification accuracy rates are calculated for each trainer based on the testing datasets. This rate measures the ability of the trainer in producing accurate results. Classification accuracy rate can be calculated as the total number of correct classification of each class over the total number of instances in the dataset. BST value is the highest classification accuracy rate that each trainer has achieved over 10 runs.

Table 3 represents AVG, STD and BST values of MSE of all trainers for all datasets. For Wisconsin breast cancer dataset, the average MSE value of GSO is 0.0400 which outperforms BAT and MVO algorithms while BBO has the lowest MSE value with 0.0386 followed by GA with 0.0345. The lowest STD was obtained by GSO algorithm by 0.0007. The best MSE value was achieved by GA based trainer with 0.0325.

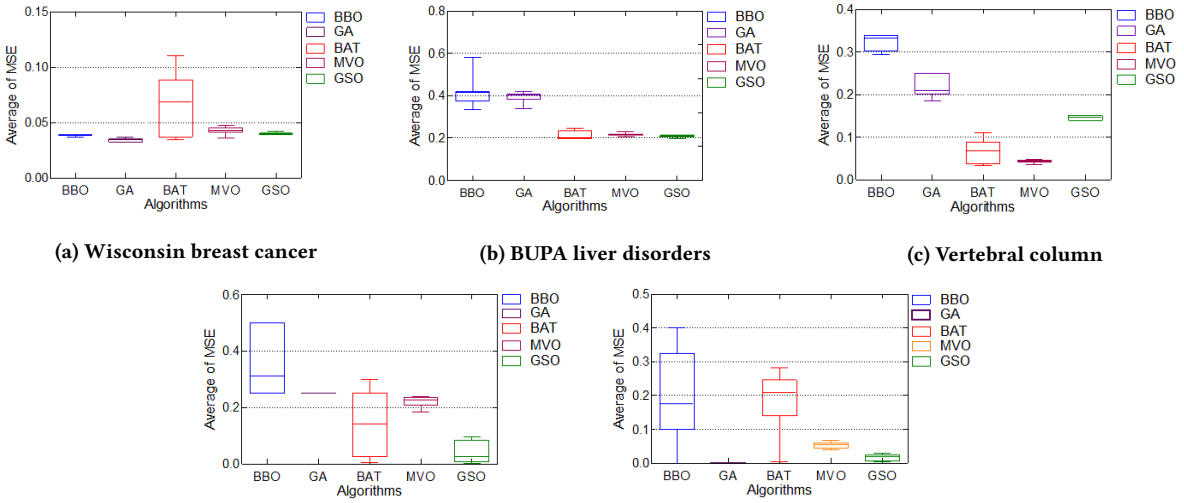
For BUPA liver disorders dataset, GSO based trainer outperforms all other trainers in terms of AVG, STD and BST values of MSE with




(a) Wisconsin breast cancer (b) BUPA liver disorders (c) Vertebral column

(d) 3-bits XOR (e) Balloons

**Figure 3: Convergence curves of algorithms in different datasets**




(a) Wisconsin breast cancer (b) BUPA liver disorders (c) Vertebral column

(d) 3-bits XOR (e) Balloons

**Figure 4: Boxplot charts of algorithms in different datasets**

0.2082, 0.0056 and 0.1966, respectively. The highest MSE value was obtained by BBO based trainer with 0.4137. For vertebral column dataset, the average MSE value of GSO is 0.1457 which also outperforms all algorithms. The lowest STD was obtained by GSO as well by 0.0054 and the highest was obtained by BAT algorithm with 0.0275. However, the best MSE value was obtained by BAT based trainer with 0.1266.

For 3-bits XOR dataset, GSO has the lowest average MSE value by 0.0381 while the highest average yields by BBO with 0.3500. However, the lowest STD value with 0 was obtained by BBO algorithm and the best MSE value was obtained by GSO based trainer with 0.0014. For balloon dataset, GA has the best results regards MSE values followed by GSO based trainer.

Table 4 represents AVG, STD and BST values of classification accuracy rates of all trainers for all datasets. For Wisconsin breast cancer dataset, the BBO based trainer is slightly better than GA and GSO algorithms with an average accuracy of 97.5%. BAT based trainer has the lowest rank with 64.5%. Moreover, GSO and BBO have obtained the lowest STD by 0.0052. The best accuracy value was obtained by BBO and GA algorithms with 98.3%.

For BUPA liver disorders dataset, GSO based trainer outperforms other trainers with an average accuracy of 72.8% while BBO based

trainer has the lowest average accuracy of 56.6%. The best accuracy rate was obtained by BAT with 77.1% and the lowest STD value was achieved by GA with 0.0129.

For vertebral column dataset, the GSO based trainer also outperforms all other trainers with an average accuracy of 83.1% followed by MVO with 79.2%. Moreover, GSO has the best classification accuracy rate as well with 89.6%. The lowest STD value was achieved by GA with 0.0217.

For 3-bits XOR dataset, the GSO based trainer competitively outperforms all algorithms with an average accuracy of 98.8% and the lowest STD of GA by 0. BBO has the lowest average accuracy of 65%. The best accuracy value of 100% was obtained by GSO and BAT algorithms. For balloon dataset, the best accuracy value of 100% was obtained by all algorithms while the highest average classification rate and lowest STD were obtained by GSO and GA algorithms.

Figures 3(a)-3(e) depict the convergence curves for all datasets employed using BBO, GA, BAT, MVO and GSO algorithms. In the convergence plots, the $x$ axis represents the number of iterations and $y$ axis represents the average MSE values over 10 runs. Figure 3(a) shows that for Wisconsin breast cancer dataset, BAT has the slowest convergence rate, while other algorithms have very small differences in the convergence rates. Figure 3(b) shows that for liver dataset, BBO followed by GA have the slowest convergence rates, while other algorithms have very small differences in the convergence rates. Figure 3(c) shows that for vertebral column dataset, GSO has the fastest convergence rate followed by MVO and BAT algorithms, while the slowest is obtained by the BBO. Figure 3(d) shows that for 3 bits XOR dataset, GA has the fastest convergence rate followed by GSO and BBO has the slowest convergence rate. Figure 3(e) shows that for balloon dataset, GA has the fastest convergence rate followed by GSO and BAT has the slowest convergence rate followed by BBO.

Figures 4(a)-4(e) depict the box-plots for all datasets employed using BBO, GA, BAT, MVO and GSO algorithms. The box-plots are used to analyse the variability in getting MSE values for 10 MSEs obtained by each trainer in the last iteration. In this plot, the box relates to the interquartile range, the whiskers represent the farthest MSEs values and the bar in the box represents the median value. The box-plots show that GSO algorithm performed well for training MLP networks.

## 6 CONCLUSION AND FUTURE WORK

This paper has proposed a new trainer to optimise the weights and biases of MLP neural network based on GSO algorithm. The objective function is to minimise the average of MSE. The performance of our proposed MLP trainer is evaluated on classification problems over five datasets: Wisconsin breast cancer, BUPA liver disorders, vertebral column, 3-bits XOR and balloons datasets. Our proposed trainer are numerically compared to four algorithms: BBO, GA, BAT and MVO.

The experimental results have showed that our proposed trainer is efficient in training MLP. According to MSE, GSO has the lowest average of MSE value in most datasets, which reflects the high local optima avoidance of this algorithm. The reason for minimum MSE of the GSO algorithm is the ability of glowworms to identify its neighbours and compute its movements by exploiting an adaptive neighbourhood, which leads to a fast convergence towards the solution.

According to the classification accuracy rate, GSO algorithm has achieved the highest rate over all other algorithms in all datasets except the Wisconsin breast cancer dataset with only 0.3% higher than the best classification value obtained by BBO based trainer. The reason for high classification rate is that GSO can balance between the exploitation and exploration. Generally, although the nature of GSO was designed for solving local optimal solution, the proposed GSO based trainer has demonstrated the efficiency of GSO in solving global optimal solution as well.

As to future work, the performance of GSO based trainer using non-parametric statistical test (i.e., Wilcoxons rank-sum test) need to be evaluated that whether our proposed algorithm presents a significant improvement over other meta-heuristic algorithms or not. In addition, efficiency of the proposed GSO in training other types of neural networks may be investigated.

## REFERENCES


[1] Oveis Abedinia and Nima Amjady. 2015. Short-term wind power prediction based on Hybrid Neural Network and chaotic shark smell optimization. *Int. Journal of Precision Engineering and Manufacturing-Green Technology* 2, 3 (2015), 245–254.

[2] Dabiah Alboaneen, Huaglory Tianfield, and Yan Zhang. 2016. Glowworm Swarm Optimisation Algorithm for Virtual Machine Placement in Cloud Computing. In *Ubiquitous Intelligence & Comp., Advanced and Trusted Comp., Scalable Comp. and Communications, Cloud and Big Data Comp., Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Intl IEEE Conf.* IEEE, 808–814.

[3] Dabiah Alboaneen, Huaglory Tianfield, and Yan Zhang. 2017. Glowworm Swarm Optimisation Based Task Scheduling for Cloud Computing. In *Proc. of the Int. Conf. on Internet of Things and Cloud Comp.,(Cambridge, 22-23 Mar)*. ACM, 1–7.

[4] Ibrahim Aljarah, Hossam Faris, and Seyedali Mirjalili. 2016. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing* (2016), 1–15.

[5] Mohammed Alweshah. 2014. Firefly algorithm with artificial neural network for time series problems. *Research Journal of Applied Sciences, Engineering and Technology* 7, 19 (2014), 3978–3982.

[6] Adrian L Arnaud, Paulo JL Adeodato, CG Vasconcelos, and Rosalvo FO. 2005. MLP neural networks optimization through simulated annealing in a hybrid approach for time series prediction. *SBC ENIA* 1110 (2005).

[7] Arthur Asuncion and David Newman. 2007. UCI machine learning repository. (2007).

[8] Ivona Brajevic and Milan Tuba. 2013. Training feed-forward neural networks using firefly algorithm. In *Proc. of the 12th Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*. 156–161.

[9] Scott E Fahlman. 1988. An empirical study of learning speed in back-propagation networks. (1988).

[10] Hossam Faris, Ibrahim Aljarah, and Seyedali Mirjalili. [n. d.]. Improved monarch butterfly optimization for unconstrained global search and neural network training. *Applied Intelligence* ([n. d.]), 1–20.

[11] Hossam Faris, Ibrahim Aljarah, and Seyedali Mirjalili. 2016. Training feedforward neural networks using multi-verse optimizer for binary classification problems. *Applied Intelligence* 45, 2 (2016), 322–332.

[12] Hossam Faris, Ibrahim Aljarah, Seyedali Mirjalili, Pedro A Castillo, and Juan J Merelo. 2016. EvoloPy: An Open-source Nature-inspired Optimization Framework in Python.. In *IJCCI (ECTA)*. 171–177.

[13] Marco Gori and Alberto Tesi. 1992. On the problem of local minima in backpropagation. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 14, 1 (1992), 76–86.

[14] Shafaatunnur Hasan, Tan Swee Quo, Siti Mariyam Shamsuddin, and Roselina Sallehuddin. 2011. Artificial Neural Network Learning Enhancement Using Artificial Fish Swarm Algorithm. In *Proc. of the 3rd Int. Conf. on computing and Informatics*. 8–9.

[15] Yu-Chi Ho and David L Pepyne. 2002. Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications* 115, 3 (2002), 549–570.

[16] John H Holland. 1992. Genetic algorithms. *Scientific american* 267, 1 (1992), 66–73.

[17] Don R Hush and Bill G Horne. 1993. Progress in supervised neural networks. *IEEE signal processing magazine* 10, 1 (1993), 8–39.

[18] Ahmad AL Kawam and Nashat Mansour. 2012. Metaheuristic optimization algorithms for training artificial neural networks. *Int. J. Comput. Inf. Technol* 1 (2012), 156–161.

[19] Erdinç Kolay, Taner Tunç, and Erol Eğrioğlu. 2016. Classification with Some Artificial Neural Network Classifiers Trained a Modified Particle Swarm Optimization. *American Journal of Intelligent Systems* 6, 3 (2016), 59–65.

[20] KN Krishnanand and Debasish Ghose. 2006. Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications. *Multiagent and Grid Sys.* 2, 3 (2006), 209–222.

[21] Youngjik Lee, Sang-Hoon Oh, and Myung Won Kim. 1993. An analysis of premature saturation in back propagation learning. *Neural networks* 6, 5 (1993), 719–728.

[22] Frank Hung-Fat Leung, Hak-Keung Lam, Sai-Ho Ling, and Peter Kwong-Shun Tam. 2003. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural networks* 14, 1 (2003), 79–88.

[23] Seyedali Mirjalili. 2015. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. *Applied Intelligence* 43, 1 (2015), 150–161.

[24] SeyedAli Mirjalili, Siti Zaiton Mohd Hashim, and Hossein Moradian Sardroudi. 2012. Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Appl. Math. Comput.* 218, 22 (2012), 11125–11137.

[25] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Abdolreza Hatamlou. 2016. Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications* 27, 2 (2016), 495–513.

[26] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. 2014. Let a biogeography-based optimizer train your multi-layer perceptron. *Information Sciences* 269 (2014), 188–209.

[27] Seyedali Mirjalili and Ali Safa Sadiq. 2011. Magnetic optimization algorithm for training multi layer perceptron. In *Communication software and networks, IEEE 3rd int. conf. on*. IEEE, 42–46.

[28] Seyedeh Zahra Mirjalili, Shahrzad Saremi, and Seyed Mohammad Mirjalili. 2015. Designing evolutionary feedforward neural networks using social spider optimization algorithm. *Neural Computing and Applications* 26, 8 (2015), 1919–1928.

[29] Satoshi Mizuta, Takashi Sato, Demelo Lao, Masami Ikeda, and Toshio Shimizu. 2001. Structure design of neural networks using genetic algorithms. *Complex Systems* 13, 2 (2001), 161–176.

[30] Udo Seiffert. 2001. Multiple layer perceptron training using genetic algorithms.. In *ESANN*. 159–164.

[31] Habib Shah, Rozaida Ghazali, and Nazri Mohd Nawi. 2011. Using artificial bee colony algorithm for MLP training on earthquake time series data prediction. *arXiv preprint arXiv:1112.4628* (2011).

[32] Paul John Werbos. 1974. Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Doctoral Dissertation, Applied Mathematics, Harvard University, MA* (1974).

[33] D. H. Wolpert and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Trans. on Evolutionary Computation* 1, 1 (Apr 1997), 67–82. https://doi.org/10.1109/4235.585893

[34] Haizhou Wu, Yongquan Zhou, Qifang Luo, and Mohamed Abdel Basset. 2016. Training Feedforward Neural Networks Using Symbiotic Organisms Search Algorithm. *Computational intelligence and neuroscience* 2016 (2016).

[35] Waleed Yamany, Mohammed Fawzy, Alaa Tharwat, and Aboul Ella Hassanien. 2015. Moth-flame optimization for training multi-layer perceptrons. In *Computer Engineering Conf., 11th Int.* IEEE, 267–272.

[36] Xin-She Yang. 2010. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization*. Springer, 65–74.

[37] Jianbo Yu, Shijin Wang, and Lifeng Xi. 2008. Evolving artificial neural networks using an improved PSO and DPSO. *Neurocomputing* 71, 4 (2008), 1054–1060.

[38] John Paul T Yusiong. 2012. Optimizing artificial neural networks using cat swarm optimization algorithm. *Int. Journal of Intelligent Systems and Applications* 5, 1 (2012), 69.

[39] Cleber Zanchettin, Teresa B Ludermir, and Leandro Maciel Almeida. 2011. Hybrid training method for MLP: optimization of architecture and training. *IEEE Transactions on Systems, Man, and Cybernetics* 41, 4 (2011), 1097–1109.